

# Analysis of Algorithms, I

## CSOR W4231

Eleni Drinea  
*Computer Science Department*

Columbia University

Randomized quicksort, balls-in-bins

**1** Randomized Quicksort

**2** Occupancy problems

# Today

**1** Randomized Quicksort

**2** Occupancy problems

# Pseudocode for randomized Quicksort

```
Randomized-Quicksort( $A, left, right$ )  
  if  $|A| = 0$  then return //  $A$  is empty  
  end if  
   $split =$  Randomized-Partition( $A, left, right$ )  
  Randomized-Quicksort( $A, left, split - 1$ )  
  Randomized-Quicksort( $A, split + 1, right$ )
```

```
Randomized-Partition( $A, left, right$ )  
   $b =$  random( $left, right$ )  
  swap( $A[b], A[right]$ )  
  return Partition( $A, left, right$ )
```

Subroutine random( $i, j$ ) returns a random number between  $i$  and  $j$  inclusive.

## Expected running time of randomized Quicksort

- ▶ Let  $T(n)$  be the **expected** running time of Randomized-Quicksort.
  - ▶ We want to bound  $T(n)$ .
  - ▶ Randomized-Quicksort differs from Quicksort only in how they select their pivot elements.
- ⇒ We will analyze Randomized-Quicksort based on Quicksort and Partition.

# Pseudocode for Partition

```
Partition(A, left, right)  
    pivot = A[right]           line 1  
    split = left - 1           line 2  
    for j = left to right - 1 do   line 3  
        if A[j] ≤ pivot then     line 4  
            swap(A[j], A[split + 1]) line 5  
            split = split + 1       line 6  
        end if  
    end for  
    swap(pivot, A[split + 1])   line 7  
    return split + 1             line 8
```

## Few observations

1. *How many times is Partition called?*

## Few observations

1. *How many times is Partition called?*

At most  $n$ .

2. Further, each Partition call spends some work

1. **outside** the for loop

2. **inside** the for loop



## Few observations

1. *How many times is Partition called?*

At most  $n$ .

2. Further, each Partition call spends some work

1. **outside** the for loop

▶ **every** Partition spends **constant** work outside the for loop

▶ at most  $n$  calls to Partition

⇒ total work **outside** the for loop in all calls to Partition is  $O(n)$

2. **inside** the for loop

# Few observations

1. *How many times is Partition called?*

At most  $n$ .

2. Further, each Partition call spends some work

1. **outside** the for loop

▶ **every** Partition spends **constant** work outside the for loop

▶ at most  $n$  calls to Partition

⇒ total work **outside** the for loop in all calls to Partition is  $O(n)$

2. **inside** the for loop

▶ let  $X$  be the total number of comparisons performed at **line 4** in **all** calls to Partition

▶ each comparison may require some further **constant** work (**lines 5 and 6**)

⇒ total work **inside** the for loop in **all** calls to Partition is  $O(X)$

## Towards a bound for $T(n)$

$X$  = the total number of comparisons in **all** Partition calls.

The running time of Randomized-Quicksort is

$$O(n + X).$$

Since  $X$  is a random variable, we need  $E[X]$  to bound  $T(n)$ .

## Towards a bound for $T(n)$

$X$  = the total number of comparisons in **all** Partition calls.

The running time of Randomized-Quicksort is

$$O(n + X).$$

Since  $X$  is a random variable, we need  $E[X]$  to bound  $T(n)$ .

### Fact 1.

*Fix any two input items. During the execution of the algorithm, they may be compared at most once.*

## Towards a bound for $T(n)$

$X$  = the total number of comparisons in **all** `Partition` calls.

The running time of `Randomized-Quicksort` is

$$O(n + X).$$

Since  $X$  is a random variable, we need  $E[X]$  to bound  $T(n)$ .

### Fact 1.

*Fix any two input items. During the execution of the algorithm, they may be compared at most once.*

### Proof.

Comparisons are only performed with the *pivot* of each `Partition` call. After `Partition` returns, *pivot* is in its final location in the output and will not be part of the input to any future recursive call.  $\square$

## Simplifying the analysis

- ▶ There are  $n$  numbers in the input, hence  $\binom{n}{2} = \frac{n(n-1)}{2}$  distinct (unordered) pairs of input numbers.
- ▶ From Fact 1, the algorithm will perform **at most**  $\binom{n}{2}$  comparisons.
- ▶ *What is the **expected** number of comparisons?*

## Simplifying the analysis

- ▶ There are  $n$  numbers in the input, hence  $\binom{n}{2} = \frac{n(n-1)}{2}$  distinct (unordered) pairs of input numbers.
- ▶ From Fact 1, the algorithm will perform **at most**  $\binom{n}{2}$  comparisons.
- ▶ *What is the **expected** number of comparisons?*

To simplify the analysis

- ▶ relabel the input as  $z_1, z_2, \dots, z_n$ , where  $z_i$  is the  $i$ -th smallest number.
- ▶ **assume** that all input numbers are **distinct**; thus  $z_i < z_j$ , for  $i < j$ .

## Writing $X$ as the sum of indicator random variables

Let  $X_{ij}$  be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$



## Writing $X$ as the sum of indicator random variables

Let  $X_{ij}$  be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$

The total number of comparisons is given by  $X = \sum_{1 \leq i < j \leq n} X_{ij}$ .

## Writing $X$ as the sum of indicator random variables

Let  $X_{ij}$  be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$

The total number of comparisons is given by  $X = \sum_{1 \leq i < j \leq n} X_{ij}$ .

$E[X] = ?$

## Writing $X$ as the sum of indicator random variables

Let  $X_{ij}$  be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$

The total number of comparisons is given by  $X = \sum_{1 \leq i < j \leq n} X_{ij}$ .

By linearity of expectation

$$E[X] = E\left[\sum_{1 \leq i < j \leq n} X_{ij}\right] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \Pr[X_{ij} = 1]$$

# Writing $X$ as the sum of indicator random variables

Let  $X_{ij}$  be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$

The total number of comparisons is given by  $X = \sum_{1 \leq i < j \leq n} X_{ij}$ .

By linearity of expectation

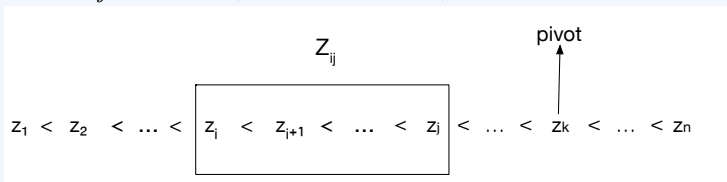
$$E[X] = E\left[\sum_{1 \leq i < j \leq n} X_{ij}\right] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \Pr[X_{ij} = 1]$$

**Goal:** compute  $\Pr[X_{ij} = 1]$ , that is, the **probability that two fixed items  $z_i$  and  $z_j$  are ever compared.**

Fix two items  $z_i$  and  $z_j$ . When are they compared?

**Notation:** let  $Z_{ij} = \{z_i, z_{i+1}, \dots, z_j\}$

Consider the initial call  $\text{Partition}(A, 1, n)$ . Assume it picks  $z_k$  **outside**  $Z_{ij}$  as *pivot* (see figure below).



1.  $z_i$  and  $z_j$  are **not** compared in this call (*why?*).
2. All items in  $Z_{ij}$  will be greater (or smaller) than  $z_k$ , so they will **all be input to the same subproblem** after  $\text{Partition}(A, 1, n)$  returns.

In the first Partition with  $pivot \in Z_{ij} = \{z_i, \dots, z_j\}$

The first Partition call that picks its *pivot* from  $Z_{ij}$  determines if  $z_i, z_j$  are ever compared. Three possibilities:

1. *pivot* =  $z_i$
2. *pivot* =  $z_j$
3. *pivot* =  $z_\ell$ , for some  $i < \ell < j$

## In the first Partition with $pivot \in Z_{ij} = \{z_i, \dots, z_j\}$

The first Partition call that picks its *pivot* from  $Z_{ij}$  determines if  $z_i, z_j$  are ever compared. Three possibilities:

1. *pivot* =  $z_i$

$z_i$  is compared with every element in  $Z_{ij} - \{z_i\}$ , thus with  $z_j$  too.  $z_i$  is placed in its final location in the output and will not appear in any future calls to Partition.

2. *pivot* =  $z_j$

$z_j$  is compared with every element in  $Z_{ij} - \{z_j\}$ , thus with  $z_i$  too.  $z_j$  is placed in its final location in the output and will not appear in any future recursive calls.

3. *pivot* =  $z_\ell$ , for some  $i < \ell < j$

$z_i$  and  $z_j$  are **never** compared (*why?*)

So  $z_i$  and  $z_j$  are compared when ...

... either of them is chosen as *pivot* in that **first** Partition call that chooses its *pivot* element from  $Z_{ij}$ .

Now we can compute  $\Pr[X_{ij} = 1]$ :

$$\Pr[X_{ij} = 1] = \Pr[z_i \text{ is chosen as } \textit{pivot} \text{ by the first Partition that picks its } \textit{pivot} \text{ from } Z_{ij}, \text{ or } z_j \text{ is chosen as } \textit{pivot} \text{ by the first Partition that picks its } \textit{pivot} \text{ from } Z_{ij}] \quad (1)$$



# The union bound

Suppose we are given a set of events  $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ , and we are interested in the probability that **any** of them happens.

**Union bound:** Given events  $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ , we have

$$\Pr \left[ \bigcup_{i=1}^n \varepsilon_i \right] \leq \sum_{i=1}^n \Pr[\varepsilon_i].$$

**Union bound for mutually exclusive events:** Suppose that  $\varepsilon_i \cap \varepsilon_j = \emptyset$  for each pair of events. Then

$$\Pr \left[ \bigcup_{i=1}^n \varepsilon_i \right] = \sum_{i=1}^n \Pr[\varepsilon_i].$$

## Computing the probability that $z_i$ and $z_j$ are compared

Since the two events in equation (1) are mutually exclusive, we obtain

$$\begin{aligned}\Pr[X_{ij} = 1] &= \Pr[z_i \text{ is chosen as } \textit{pivot} \text{ by the first Partition} \\ &\quad \text{call that picks its } \textit{pivot} \text{ from } Z_{ij}] \\ &+ \Pr[z_j \text{ is chosen as } \textit{pivot} \text{ by the first Partition} \\ &\quad \text{call that picks its } \textit{pivot} \text{ from } Z_{ij}] \\ &= \frac{1}{j-i+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1},\end{aligned}\tag{2}$$

since the set  $Z_{ij}$  contains  $j-i+1$  elements.

From  $\Pr[X_{ij} = 1]$  to  $E[X]$

$$\begin{aligned} E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \Pr[X_{ij} = 1] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j-i+1} \\ &= 2 \sum_{i=1}^{n-1} \sum_{\ell=2}^{n-i+1} \frac{1}{\ell} \end{aligned} \quad (3)$$

Note that  $\sum_{\ell=1}^k \frac{1}{\ell} = H_k$  is the  **$k$ -th harmonic number**, such that

$$\ln k \leq H_k \leq \ln k + 1 \quad (4)$$

Hence  $\sum_{\ell=2}^{n-i+1} \frac{1}{\ell} \leq \ln(n-i+1)$ . Substituting in (3), we get

$$E[X] \leq 2 \sum_{i=1}^{n-1} \ln(n-i+1) \leq 2 \sum_{i=1}^{n-1} \ln n = O(n \ln n)$$

## From $E[X]$ to $T(n)$

- ▶ Equations (3), (4) also yield a lower bound of  $\Omega(n \ln n)$  for  $E[X]$  (*show this!*).
- ▶ Hence  $E[X] = \Theta(n \ln n)$ . Then the expected running time of **Randomized-Quicksort** is

$$T(n) = \Theta(n \ln n)$$

# Today

1 Randomized Quicksort

2 Occupancy problems

# Balls in bins problems

**Occupancy problems:** find the distribution of balls into bins when  $m$  balls are thrown independently and uniformly at random into  $n$  bins.

- ▶ Applications: analysis of randomized algorithms and data structures (e.g., **hash table**)

*Q1: How many balls can we throw before it is more likely than not that some bin contains at least two balls?*

In symbols: *find  $k$  such that*

$$\Pr[\exists \text{ bin with } \geq 2 \text{ balls after } k \text{ balls thrown}] > 1/2$$

## Easier to analyze the complement of this event

Easier to think about the probability of the complementary event.

*Q1 (rephrased): Find  $k$  such that*

$$\Pr[\mathbf{every} \text{ bin has } \leq 1 \text{ ball after } k \text{ balls thrown}] \leq 1/2$$

## Analysis: one ball at a time

- ▶ The 1st ball falls into some bin.
- ▶ The 2nd ball falls into a new bin w. prob.  $1 - \frac{1}{n}$ .
- ▶ The 3rd ball falls into a new bin (given that the first two balls fell into different bins) w. prob.  $1 - \frac{2}{n}$ .
- ▶ The  $m$ -th ball falls into a new bin (given that the first  $k - 1$  balls fell into different bins) w. prob.  $1 - \frac{k-1}{n}$ .

By the chain rule of conditional probability, the probability that the  $k$ -th ball falls into a new bin is given by

$$\prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right) \tag{5}$$



## Application: the birthday paradox

Use  $1 + x \leq e^x$  for all  $x \geq 0$  to upper bound (5)

$$\prod_{i=1}^{k-1} e^{-i/n} = e^{-\sum_{i=1}^{k-1} i/n} = e^{-\frac{k(k-1)}{(2 \cdot n)}} \approx e^{-\frac{k^2}{2n}} \quad (6)$$

Requiring  $e^{-\frac{k^2}{2n}} < 1/2$  yields  $k > \sqrt{n \cdot 2 \ln 2} = \Omega(\sqrt{n})$ .

► **Application:** birthday paradox

*Assumption: For  $n = 365$ , each person has an independent and uniform at random birthday from among the 365 days of the year.*

Once 23 people are in a room, it is more likely than not that two of them share a birthday.

## More balls-in-bins questions

- ▶ *Q2: What is the expected load of a bin after  $m$  balls are thrown?*
- ▶ *Q3: What is the expected #empty bins after  $m$  balls are thrown?*
- ▶ *Q4: What is the load of the fullest bin **with high probability**?*
- ▶ *Q5: What is the expected number of balls until **every** bin has at least one ball (Coupon Collector's Problem)?*

## Expected load of a bin

Suppose that  $m$  balls are thrown independently and uniformly at random into  $n$  bins. Fix a bin  $j$ .

- ▶ Let  $X_{ij}$  be an indicator r.v. such that  $X_{ij} = 1$  if and only if ball  $i$  falls into bin  $j$ . Then

$$E[X_{ij}] = \Pr[X_{ij} = 1] = \frac{1}{n}.$$

The total #balls in bin  $j$  is given by  $X_j = \sum_{i=1}^m X_{ij}$ . By linearity of expectation,

$$E[X_j] = \sum_{i=1}^m E[X_{ij}] = m/n.$$

Since bins are symmetric, the expected load of any bin is  $m/n$ .

## Expected # empty bins

Suppose that  $m$  balls are thrown independently and uniformly at random into  $n$  bins. Fix a bin  $j$ .

- ▶ Let  $Y_j$  be an indicator r.v. such that  $Y_j = 1$  if and only if bin  $j$  is empty.
- ▶  $\Pr[\text{ball } i \text{ does not fall in bin } j] = 1 - 1/n$
- ▶  $\Pr[\text{for all } i, \text{ ball } i \text{ does not fall in bin } j] = (1 - 1/n)^m$
- ▶ Hence  $\Pr[Y_j = 1] = (1 - 1/n)^m$ .

The number of empty bins is given by the random variable  $Y = \sum_{j=1}^n Y_j$ . By linearity of expectation

$$E[Y] = \sum_{j=1}^n E[Y_j] = \left(1 - \frac{1}{n}\right)^m \approx ne^{-m/n}$$

# Maximum load with high probability (case $m = n$ )

## Proposition 1.

*When throwing  $n$  balls into  $n$  bins uniformly and independently at random, the maximum load in any bin is  $\Theta(\ln n / \ln \ln n)$  with probability close to 1 as  $n$  grows large.*

## Two-sentence sketch of the proof.

1. Upper bound the probability that **any** bin contains more than  $k$  balls by a union bound: 
$$\sum_{j=1}^n \sum_{\ell=k}^n \binom{n}{\ell} \left(\frac{1}{n}\right)^\ell \left(1 - \frac{1}{n}\right)^{n-\ell}.$$
2. Compute the smallest possible  $k^*$  such that the probability above is less than  $1/n$ ; the latter becomes negligible as  $n$  grows large.



## Expected #balls until no empty bins

Suppose that we throw balls independently and uniformly at random into  $n$  bins, one at a time (the first ball falls at time  $t = 1$ ).

- ▶ We call a throw a **success** if it lands in an empty bin.
- ▶ We call the sequence of balls starting after the  $(j - 1)$ -st success and ending with the  $j$ -th success, the  $j$ -th **epoch**.
- ▶ Clearly the first ball is a **success**, hence ends epoch 1.
- ▶ Let  $\eta_2$  be the #balls thrown in epoch 2.

$$\forall t \in \text{epoch 2, } \Pr[\text{ball } t \text{ in epoch 2 is a success}] = \frac{n-1}{n}$$

- ▶ Similarly, let  $\eta_j$  be the #balls thrown in epoch  $j$ .

$$\forall t \in \text{epoch } j, \Pr[\text{ball } t \text{ in epoch } j \text{ is a success}] = \frac{n-j+1}{n}$$

At the end of the  $n$ -th epoch, each of the  $n$  bins has at least one ball.

## Expected #balls until no empty bins (cont'd)

Let  $\eta = \sum_{j=1}^n \eta_j$ . We want

$$E[\eta] = E \left[ \sum_{j=1}^n \eta_j \right] = \sum_{j=1}^n E[\eta_j]$$

- ▶ Each epoch is geometrically distributed with success probability  $p_j = \frac{n-j+1}{n}$ .
- ▶ Recall that the expectation of a geometrically distributed variable with success probability  $p$  is given by  $1/p$ .
- ▶ Thus  $E[\eta_j] = \frac{1}{p_j} = \frac{n}{n-j+1}$ .

Then

$$E[\eta] = \sum_{j=1}^n \frac{n}{n-j+1} = n \sum_{j=1}^n \frac{1}{j} = n(\ln n + O(1))$$

# Probability review

- ▶ A sample space  $\Omega$  consists of the possible outcomes of an experiment.
- ▶ Each point  $x$  in the sample space has an associated probability mass  $p(x) \geq 0$ , such that  $\sum_{x \in \Omega} p(x) = 1$ .
- ▶ **Example experiment: flip a fair coin;**  
 $\Omega = \{heads, tails\}; \Pr[heads] = \Pr[tails] = 1/2$ .
- ▶ We define an event  $\mathcal{E}$  to be any subset of  $\Omega$ , that is, a collection of points in the sample space.
- ▶ We define the probability of the event to be the sum of the probability masses of all the points in  $\mathcal{E}$ . That is,

$$\Pr[\mathcal{E}] = \sum_{x \in \mathcal{E}} p(x)$$